Mosaic-IT: Cost-Free Compositional Data Synthesis for Instruction Tuning

Ming Li[1], **Pei Chen**[2], **Chenguang Wang**[3], **Hongyu Zhao**[1], **Yijun Liang**[1]
Yupeng Hou[2], **Fuxiao Liu**[1], **Tianyi Zhou**[1]
[1]University of Maryland [2]Texas A&M University [3]Stony Brook University
minglii@umd.edu tianyi.david.zhou@gmail.com

Abstract

Finetuning large language models with a variety of instruction-response pairs has enhanced their capability to understand and follow instructions. Current instruction tuning primarily relies on teacher models or human intervention to generate and refine the instructions and responses for training, which are costly, non-sustainable, and may lack diversity. In this paper, we introduce Mosaic Instruction Tuning (Mosaic-IT), a human/model-free compositional data synthesis method that can efficiently create rich and diverse augmentations from existing instruction tuning data to enhance the LLMs. Mosaic-IT randomly concatenates multiple instruction data into one and trains the model to produce the corresponding responses with predefined higher-level meta-instructions to strengthen its multi-step instruction-following and format-following skills. Our extensive evaluations demonstrate a superior performance and training efficiency of Mosaic-IT, which achieves consistent performance improvements over various benchmarks and an 80% reduction in training costs compared with original instruction tuning. Our codes and data are available at https://github.com/tianyi-lab/Mosaic-IT.

1 Introduction

The emergence of Large Language Models (LLMs) (Scao et al., 2022; OpenAI, 2023; Touvron et al., 2023a) along with their remarkable performance in downstream tasks (Zhao et al., 2023; Xu et al., 2024a), has revolutionized the domains of Artificial Intelligence and Natural Language Processing. A key component of the recipe to unlock the exceptional ability of LLMs in understanding and following instructions is the technique of Instruction Tuning (IT) (Mishra et al., 2021; Wei et al., 2022; Chung et al., 2022), which involves the fine-tuning of LLMs on datasets comprising corresponding instruction-response pairs.

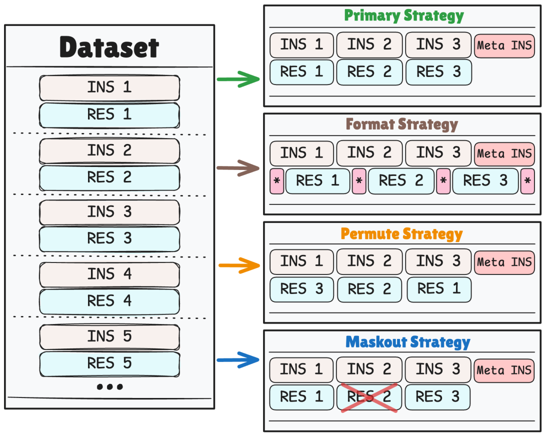


Figure 1: The brief illustration of our Mosaic-IT with different strategies. Given the original dataset, our method randomly samples and concatenates them together into more complex samples, simulating the multi-instruction-following scenarios *in a cost-free manner*.

To ensure the quality of instruction tuning data, earlier efforts (Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023a; Jiang et al., 2023) carefully curate extensive, diverse, and high-quality datasets manually. Although these datasets encompass a wide range of instructions to improve instruction tuning, they require the responses to be meticulously curated by human experts (Khashabi et al., 2020; Ye et al., 2021; Wei et al., 2022; Wang et al., 2022; Du et al., 2022). Alternatively, some approaches (Wang et al., 2023b; Taori et al., 2023; Xu et al., 2023; Li et al., 2023a) leverage more capable teacher LLMs to reduce the labor-intensive process of data generation. For example, the Alpaca (Taori et al., 2023) utilizes self-instruct (Wang et al., 2023b) to automatically generate diverse instruction tuning datasets. Building on this trend and the widely acknowledged notion that more complicated instructions are more beneficial for LLMs' instruction-following ability (Xu et al., 2023; Zhao et al., 2024), numerous strategies (Zhao et al., 2024; Wu et al., 2024; Ding et al., 2023; Li et al., 2023a; Liu et al., 2023a; Li et al., 2024c,b;

Findings of the Association for Computational Linguistics: ACL 2025, pages 25287–25318
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

Guo et al., 2024; Xu et al., 2024a) have been proposed to further diversify and complexity the instruction-response pairs, utilizing teacher models like ChatGPT-3.5 and GPT-4 (OpenAI, 2023).

Despite the enhanced performance in instruction-following ability offered by these existing methods, they face **Two** major issues: (1) They heavily rely on teacher models or human annotators to rewrite instruction-response pairs, which highlights the resource-intensive nature and their constraints on scalability; (2) They only increase the complexity within the scope of a single instruction, which limits the potential improvement in LLMs' instruction-following capabilities. Motivated by the Dense and Aligned Captions (Doveh et al., 2023) proposed for vision language (VL) models and the mosaic data augmentation proposed in Yolov4 (Bochkovskiy et al., 2020), we hypothesize that denser instructions benefit the LLM alignment, i.e. the process of instruction tuning should not be constrained by one single instruction but be extended to *follow several instructions at a time*, which represents a higher level of instruction-following ability that is beneficial to the training process. A similar concept during the inference phase is proposed by batch prompting (Cheng et al., 2023; Lin et al., 2024), where multiple samples are grouped in one batch, allowing LLMs to generate multiple responses at one inference, while its performance is sub-optimal.

As orthogonal to the existing instruction tuning methods, we introduce Mosaic Instruction Tuning (Mosaic-IT), an innovative and model/human-free compositional approach that augments existing instruction tuning datasets, which concurrently improves the LLM performances and lowers the training expenses. As shown in Figure 1, in our method, multiple instructions and corresponding responses from the original dataset are concatenated into a single sample for fine-tuning, simulating the multi-instruction-following scenarios *at no cost*. Without applying any additional strategies, we term this simple process as the **Primary Mosaic Strategy**. We posit that this mosaic strategy process significantly improves the complexity and density of the original instructions, learning from which directly benefits LLMs in their instruction-following ability. Additionally, this method offers the advantage of directly reducing the total count of instruction-response pairs, thereby cutting down on training iterations, and accelerating the training process significantly by approximately 80% reduction.

Though effective, the Primary Mosaic strategy constrains LLMs in responding to the instructions in the original order and format, potentially limiting their further potential. Thus, we further introduce three **Advanced Mosaic Strategies** aimed at enhancing the diversity and complexity of the mosaicked instruction-response pairs: **Format**, **Permute**, and **Maskout**, in which an additional meta-instruction is provided as a higher-level guideline for LLMs to follow the given instructions. Illustrative examples are presented in Figure 2. Specifically, in the Format strategy, some arbitrary parsing formats will be defined in the meta-instruction, thus forcing LLMs to follow these formats, which notably enhances the LLMs' capacity to follow formats. In the Permutation strategy, an arbitrary permuted order is defined, thus forcing LLMs to respond in a desired order. In the Maskout strategy, some arbitrary instructions are sampled, which meta-instruction forces LLMs to ignore. Moreover, the use of these Advanced strategies not only boosts the performance in several evaluation metrics but also keeps our method free of additional costs.

In summary, our primary contributions can be illustrated as follows:

- We propose the *first cost-free data Synthesis method*, **Mosaic-IT**, which extends existing instruction tuning from handling one single instruction at a time to following multiple instructions in diverse forms. This approach significantly enhances the potential utilization of existing high-quality datasets.
- Mosaic-IT improves the instruction-following abilities of LLMs compared to training on original data, as evidenced by consistent performance gains across a wide range of benchmarks, model families, and datasets, demonstrating strong generalization capabilities.
- Mosaic-IT substantially increases training efficiency by reducing the required number of training iterations, resulting in an approximate 80% reduction in training time, as confirmed by experimental results.

2 Methodology

2.1 Preliminaries

The instruction tuning dataset, defined as D, consists of n data samples, each represented by a triplet $(Instruction, Input, Response)$. For simplicity, we define $x = map(Instruction, Input)$ as

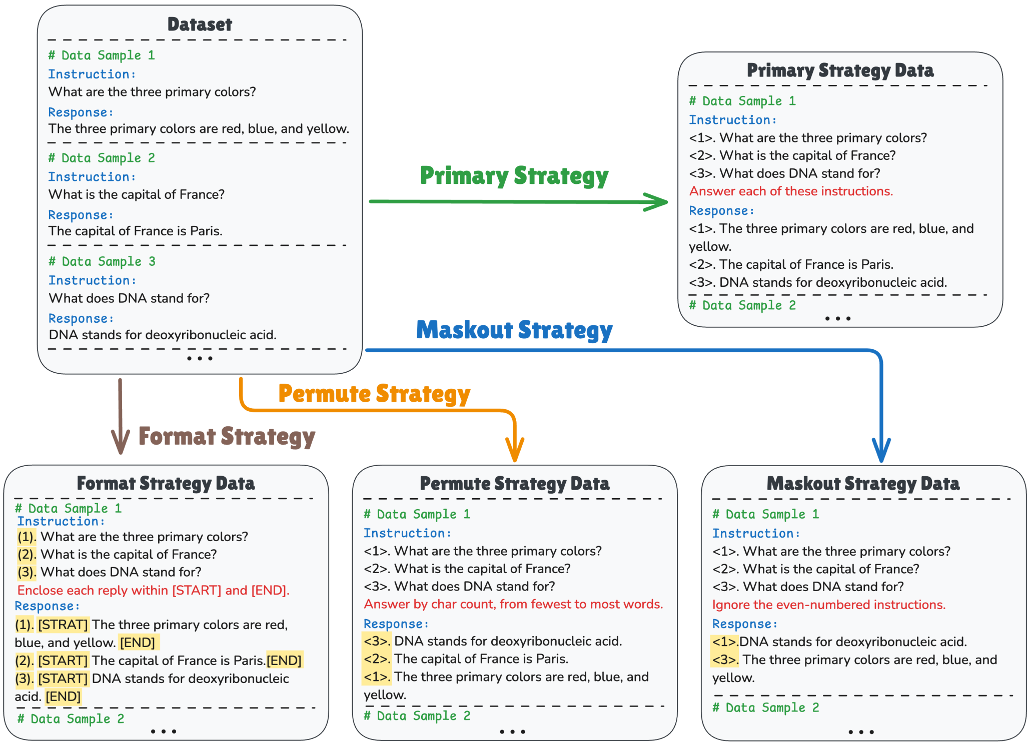


Figure 2: Illustrative examples of Mosaic-IT. Given 3 simple data instances, our method concatenates them into data samples with diverse forms. Texts in red represent the meta-instructions that define the formats or orders for LLMs to respond. Texts in yellow are major response differences of each strategy. The **Primary Strategy** only concatenates data. The **Format Strategy** requires LLMs to respond in predefined formats. The **Permute Strategy** requires LLMs to respond in specific orders, and the **Maskout Strategy** requires LLMs to ignore some instructions.

the unified instruction, and y as the corresponding response. Therefore, D can be represented as $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, denoting a set of n instruction-response pairs. Let $p_\theta(\cdot)$ denote the LLMs to be trained, with parameters θ. In the instruction tuning setting, p_θ is typically fine-tuned by maximizing the following objective on each data (x_i, y_i), $y_{i,j}$ represents the j_{th} token of response y_i, $y_{i,<j}$ represents the tokens prior to $y_{i,j}$, and l_i represents the token length of $y_{i,j}$:

$$\max_{\theta} \sum_{j=1}^{l_i} \log p_\theta \left(y_{i,j} | x_i, y_{i,<j} \right), \tag{1}$$

2.2 Mosaic-IT

Motivated by the success of the existing data-centric instruction tuning methods, a line of approaches is proposed to further enhance the instruction-response pairs utilizing extra teacher LLMs (Xu et al., 2024a). Though effective, all existing methods for instruction tuning restrict training samples to just one instruction, which severely limits the potential of the existing high-quality data and the instruction-following ability of the models to be trained. Motivated by the Dense and Aligned Captions (Doveh et al., 2023) for VL, we hypothesize that denser instructions benefit the LLM alignment, thus the process of instruction tuning should not be constrained by one single instruction but be extended to follow several instructions at a time, which represents a higher level of instruction-following ability that is beneficial to the training process. Thus, we propose the cost-free data synthesis method, Mosaic Instruction Tuning (Mosaic-IT), as shown in Figure 1.

2.2.1 Primary Mosaic strategy

Exploring the concept of concatenating random instruction-response pairs into a unified instruction-response pair for training remains largely unexplored. The primary challenge lies in crafting a coherent overall instruction and obtaining its corresponding response. Most existing methods utilize a strong teacher model to rewrite the instructions with prompting techniques and generate corresponding responses, introducing more cost by actually re-generating new data samples. To harness the full potential of existing data rather than directly discarding it, we introduce a simple cost-free compositional approach as shown in Figure 2, in which instructions are randomly concatenated with serial digits to form an *overall instruction*. The concatenated overall instruction is denoted as $[x_1, ..., x_k]$, with the corresponding overall response concatenated as $[y_1, ..., y_k]$. Here, k denotes the number of original data samples integrated into each overall sample.

In this framework, the fundamental instruction-following capability is triggered by the existing instruction-response pairs, and the mosaic strategy extends this capability to a higher level in which LLMs are forced to follow multiple instructions. It represents a much more complicated scenario that benefits LLMs compared with traditional single-task instructions. Consequently, the objective function for each concatenated overall data sample can be formulated as follows:

$$\max_{\theta} \sum_{j=1}^{l} \log p_{\theta} \left([y_1, ..., y_k]_j | [x_1, ..., x_k], [y_1, ..., y_k]_{<j} \right), \tag{2}$$

Here, $[y_1, ..., y_k]_j$ denotes the jth token of the overall response, $[y_1, ..., y_k]_{<j}$ denotes the tokens prior to jth token, and l represents the length of overall response. This formulation encapsulates the essence of our approach, optimizing the model parameters θ to maximize the likelihood of generating the correct sequence of responses for the given overall instruction.

2.2.2 Advanced Mosaic Strategies

Though effective, this primary mosaic strategy constrains LLMs in responding to the instructions with the original order and format, potentially limiting their generalization and practical usage. In our method, the instructions and corresponding responses from the original dataset can be viewed as atomic components, and our method randomly combines these elements to form new instructions and responses. This nature allows us to further complicate this process with fancier strategies, thus forcing LLMs to follow more complicated overall instructions. Hence, we propose three **Advanced Mosaic Strategies** to complicate and diversify the mosaicked samples as shown in Figure 2, including Format, Permute, and Maskout, with meta-instructions guiding them. These strategies are still purely rule-based (Li et al., 2024a) and do not incorporate the additional human/LLM generation.

Format In the Format strategy, some arbitrary formats are defined in the meta-instruction to force LLMs to follow these formats in the response. The formats mainly contain two categories: 1) *Serial Digit Format* and (2) *Response Parsing Format*. The serial digits establish the initial instruction order that guides LLMs to follow sequentially. We manually define 10 types of serial digit format, which will be randomly sampled during each mosaic process. For response parsing, we simulate the scenario where the users try to extract specific information from the responses. We define 27 types of parsing brackets and 17 types of parsing text pairs, which will be randomly sampled and assembled during each mosaic process. Examples can be found in Appendix G, which can be easily extended for customized training settings. We denote responses with specific formats as $y_i' = wrap(y_i, s_{format})$, and l as the token length of the overall response. An additional meta-instruction s_{format} specifying the required format will be included in the overall instruction. Thus, the objective function for each mosaic data point:

$$\max_{\theta} \sum_{j=1}^{l} \log p_{\theta} \Big([y_1', ..., y_k']_j \Big| [x_1, x_2, ..., x_k, s_{format}], [y_1', ..., y_k']_{<j} \Big) \tag{3}$$

Permute and Maskout Building upon the Format strategy, we further introduce two strategies for our Mosaic-IT, Permutation and Maskout.

In the **Permute** strategy, an arbitrary permuted order is defined in the meta-instructions, forcing LLMs to follow. Moreover, several high-level rules are defined to ensure the complexity and diversity of meta-instructions, e.g., forcing LLMs to respond to each instruction in the randomly generated permutation list, forcing LLMs to respond in the alphabetical order of each instruction, forcing LLMs to respond according to the length of instructions, etc. The detailed rule types and descriptions are depicted in Appendix G. These various meta-instructions not only provide higher-level guidelines for LLMs to follow multiple instructions but also inherently enhance the instruction perception ability of LLMs. In our settings, LLMs are required to generate responses selectively conditioned on some critical parts of the overall instruction, forcing them to first understand the formats and other requirements, indicating a more comprehensive understanding of the context given. The meta-instruction is denoted as $s_{permute}$ and is included in the overall instruction. The permuted response list is denoted as $[y_{1'}', ..., y_{k'}'] = Permute([y_1', ..., y_k'], s_{permute})$. Thus, the objective function can be formulated as below:

$$\max_{\theta} \sum_{j=1}^{l} \log p_{\theta}\Big([y'_{1'}, ..., y'_{k'}]_j \Big| [x_1, \quad (4)$$
$$x_2, ..., x_k, s_{format}, s_{permute}], [y'_{1'}, ..., y'_{k'}]_{<j}\Big)$$

In the **Maskout** strategy, some arbitrary instructions are selected in the meta-instructions, forcing LLMs to ignore them. Several high-level rules are also defined similarly to the permute strategy, including forcing LLMs to ignore the instructions with given random digits, forcing LLMs to ignore the longest one/several instructions, forcing LLMs to ignore odd-numbered instructions, etc. The details are provided in Appendix G. Similarly, the meta-instruction is denoted as $s_{maskout}$ and the response list is denoted as $[y'_1, ..., y'_m] = Maskout([y'_1, ..., y'_k], s_{maskout})$, where m is the count of responses after masking out. Thus, the objective function can be formulated as below:

$$\max_{\theta} \sum_{j=1}^{l} \log p_{\theta}\Big([y'_1, ..., y'_m]_j \Big| [x_1, \quad (5)$$
$$x_2, ..., x_k, s_{format}, s_{maskout}], [y'_1, ..., y'_m]_{<j}\Big)$$

It's important to note that our mosaic strategies entail *no supervision cost*, and the predefined rules are flexible and have the potential for further extension. We utilize the version with three Advanced strategies as our default Mosaic-IT.

How to decide the Number of Instructions k: Number of Instructions denotes the number of original data samples that are integrated into an overall sample. In addition to the detailed mosaic strategies being used, this count also dramatically affects the effect of Mosaic-IT. Our experiments reveal that larger and more diverse numbers of instructions will benefit LLM training. By default, we set the maximum number of instructions as $k_{max} = 10$, and randomly sample an integer that is smaller than or equal to k_{max} under a uniform distribution. If the number causes the data sample to be longer than the max length, it will be automatically reduced to the max number, which remains the sample length within the limits.

3 Experimental Results

3.1 Main Results

In this section, we present the evaluation results comparing our methods with the baseline methods on ***6*** baseline models (Mistral-7B (Jiang et al., 2023), Llama2-7B (Touvron et al., 2023b), Llama2-13B, Llama-3-8B (Dubey et al., 2024), Phi-3 (Abdin et al., 2024), Gemma2-2B (Team et al., 2024)) and ***4*** instruction tuning datasets (Alpaca-GPT4 (Peng et al., 2023), Alpaca (Taori et al., 2023), WizardLM-70k (Xu et al., 2023)), Magpie (Xu et al., 2024b), on ***5*** commonly used evaluation metrics and additional Human Evaluation. Detailed experimental setup and descriptions of evaluation metrics can be found in Appendix B.

Table 1 shows the results on 2 general evaluation settings (Pair-Wise Comparison and Open LLM leaderboard). **Pair-wise Winning Score** indicates the result directly comparing our models with the corresponding baseline models, which is calculated as *(Num(Win)−Num(Lose))/Num(All)* $+1$. These values that are greater than 1.0 represent better responses generated by our models. The performances on the **Huggingface Open LLM Leaderboard** are also presented, and we bold the greater average values for each comparison. The consistently outperforming results on different base models and datasets represent the effectiveness and robustness of our methods. Results on more advanced baseline models and datasets can be found in Table 2. Our method shows consistent improvements compared with baseline models. The performance gains in pair-wise comparison indicate our method helps LLMs generate more detailed and accurate responses, and the performance gains in the open leaderboard indicate our method helps alleviate harm to the intrinsic capabilities of base LLMs.

To better understand how our method improves the instruction-following abilities of LLMs, we further compare the performance on the other 3 benchmarks for fine-grained analysis as shown in Table 2. On the **Alpaca Eval 2** benchmark, our method has a consistent improvement with or without the Length Control (LC), indicating that the improvement of response qualities does not directly originate from the length of responses. On the **MT-Bench**, the 1-round scores of our method are consistently higher, while the 2-round scores slightly fluctuate, indicating that our method mainly improves the response quality for single-round conversations since our meta instructions only focus on single-round scenarios in this version. On the **IF Eval** benchmark, our method consistently improves the performance on different settings, both Prompt-level and Instruction-level. Compared with the previous benchmarks, IF Eval mainly focuses on the constraint-following ability of LLMs. The

Model	Dataset	Method	Pair-wise ↑ Winning Score	Huggingface Open LLM Leaderboard ↑				
				Average	ARC	HellaSwag	MMLU	TruthfulQA
Mistral-7B	Alpaca-GPT4	Baseline	1.000	59.70	55.03	78.87	56.01	48.88
		Mosaic-IT	**1.349**	**63.65**	59.04	81.85	60.09	53.62
	Alpaca	Baseline	1.000	55.15	51.96	74.61	52.85	41.20
		Mosaic-IT	**1.390**	**58.86**	56.23	79.57	57.06	42.58
	Wizard-70k	Baseline	1.000	57.86	51.88	77.93	53.76	47.89
		Mosaic-IT	**1.161**	**61.11**	57.85	82.13	57.42	47.08
Llama2-7B	Alpaca-GPT4	Baseline	1.000	58.71	54.69	80.05	47.89	52.21
		Mosaic-IT	**1.073**	**58.84**	54.18	80.54	47.92	52.70
	Alpaca	Baseline	1.000	55.25	54.35	78.65	47.02	40.98
		Mosaic-IT	**1.096**	**55.32**	53.75	78.65	46.88	41.98
	Wizard-70k	Baseline	1.000	57.09	54.18	79.25	46.93	48.02
		Mosaic-IT	**1.197**	**57.41**	54.69	79.69	48.11	47.13
Llama2-13B	Alpaca-GPT4	Baseline	1.000	61.47	58.70	83.12	54.13	49.92
		Mosaic-IT	**1.110**	**63.26**	58.87	83.54	55.75	54.87
	Alpaca	Baseline	1.000	57.63	57.25	81.23	54.13	37.91
		Mosaic-IT	**1.046**	**58.80**	56.57	81.79	54.28	52.55
	Wizard-70k	Baseline	1.000	61.24	57.04	83.39	55.76	48.78
		Mosaic-IT	**1.078**	**61.50**	58.70	83.69	56.44	47.18

Table 1: The performance comparison on the Pair-wise Comparison Winning Score and the Open LLM Leaderboard, on 3 different base models and 3 different instruction tuning datasets.

Model	Dataset	Method	Pair-wise ↑ Score	Open LLM ↑ Average	Alpaca Eval 2 ↑		MT-Bench ↑		IF Eval ↑	
					Rate (LC)	Rate	1-round	2-round	P(L)	I(L)
Llama-3-8B	Magpie	Baseline	1.000	56.15	9.22	13.74	8.10	7.08	35.67	47.72
		Mosaic-IT	**1.133**	**60.13**	**12.23**	**16.05**	**8.36**	**7.49**	**40.67**	**52.76**
Phi-3	Magpie	Baseline	1.000	62.90	13.82	**17.68**	7.78	**6.42**	44.36	55.52
		Mosaic-IT	**1.014**	**63.54**	**14.04**	17.67	**7.89**	6.16	**50.83**	**62.35**
Gemma2-2B	Magpie	Baseline	1.000	46.37	5.35	7.77	4.57	3.23	21.81	32.49
		Mosaic-IT	**1.032**	**48.36**	**5.66**	**8.54**	**5.16**	**3.96**	**22.18**	**34.77**
Mistral 7B	Alpaca-GPT4	Baseline	1.000	59.70	3.98	7.28	6.44	**5.26**	35.86	45.92
		Mosaic-IT	**1.349**	**63.65**	**5.00**	**7.81**	**7.11**	4.69	**38.08**	**50.23**
	Wizard-70k	Baseline	1.000	57.86	4.13	6.46	6.21	**4.70**	41.96	53.00
		Mosaic-IT	**1.161**	**61.11**	**4.44**	**7.56**	**6.95**	4.32	**45.47**	**56.47**

Table 2: The performance comparison across multiple model families and datasets on five evaluation metrics. Rate(LC) in Alpaca Eval represents length-controlled win rates. In IF Eval, P(L) and I(L) represent Prompt-level and Instruction-level accuracy in the Loose setting.

consistent improvement in this benchmark represents that our method not only improves the response qualities of the LLMs but also improves their controllability regarding formats.

Further **Human Evaluations** are conducted on Mistral-7B with the Alpaca-GPT4 and WizardLM datasets. For the comparison on (1) Alpaca-GPT4: the model using Mosaic-IT wins on 68 out of 100 instruction, ties on 3, and losses on 29 instructions; on (2) WizardLM: the model using Mosaic-IT wins on 63 out of 100 instruction, ties on 6, and losses on 31 instructions. This human evaluation also further verifies the effectiveness of our Mosaic-IT.

To conclude, our Mosaic-IT shows consistent improvement in instruction-following and constraint-following ability and response quality with a reduction of approximately 80% of the training time cost. *Given that our method is a cost-free data synthesis technique that does not rely on any additional human/LLM generation, the observed improvements are remarkable.*

3.2 Ablation Studies

Detailed ablations are in Appendix C.

Ablation on Mosaic Strategies investigates how different mosaic strategies affect LLM performances. We find out that further implementing Advanced Strategies (Format, Permute, Maskout) improves LLM performance as they largely diversify and complicate the instructions, only implementing the Primary Strategy. **Ablation on the Max Number of Instructions** investigates the trend between the number of instructions that are concatenated

Settings	Baseline	Fix	Exponential	Pareto	Log-normal	Logistic	Uni-2	Uni-4	Uni-6	Uni-8	Uni-10	Uni-12
Time (min)	827	121	129	133	133	143	716	426	305	245	202	173
Time Ratio	100.0%	14.6%	15.6%	16.1%	16.1%	17.3%	86.6%	51.5%	36.9%	29.6%	24.4%	20.9%
Winning Score	1.000	0.982	0.995	1.417	1.431	1.417	0.989	1.142	1.303	1.294	1.349	1.376

Table 3: The training time comparison of different settings, and the pair-wise winning scores are also provided for better illustration. "Uni-2" represents uniform distribution with max count as 2. **Mosaic-IT reduces the training time to $16\% - 25\%$ while achieving better performance.**

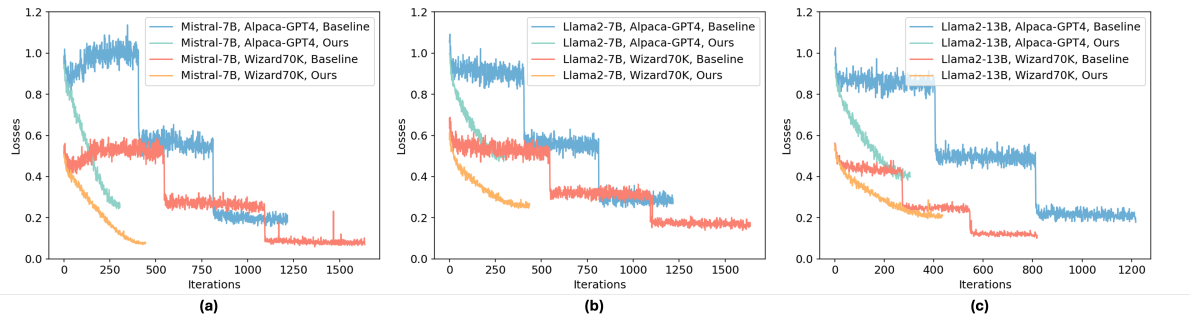


Figure 3: The training loss curve comparisons between the original instruction tuning process and our Mosaic-IT with w datasets on (a) Mistral-7B, (b) Llama2-7B, and (c) Llama2-13B. **The "stair-like" loss curves for the original training process indicate potential memorizing effects, while our loss curves are smoother.** All the training settings are kept the same between the baseline models and Mosaic-IT models, including the Learning Rate, Warm-up Ratio, Learning Rate Schedule (Cosine), Batch Size, etc.

together and the LLMs' performances. We find out that in the uniform distribution setting, more of the instructions are concatenated together, as this process makes the concatenated instruction complex. **Ablation on the Distribution of Number of Instructions** investigates how different distributions affect the LLMs' performance. It is revealed that, except for the maximum number of instructions concatenated together, the distribution is also important. **Ablation on Semantic Grouping** investigates how semantic grouping, i.e., grouping instructions with similar semantic meaning rather than pure random grouping, affects performance. We show that semantic grouping has its own benefits, while random selection is much more convenient.

4 Further Discussion

4.1 Improving Efficiency

One of the benefits of our method is the efficiency of the training process. Given an existing dataset, our mosaic processes largely decrease the number of total overall instructions and the total number of gradient descents, leading to a reduction in the training process. The detailed comparison is shown in Table 3, which is based on the Mistral-7B model on the Alpaca-GPT4 dataset. The time is calculated based on four NVIDIA A100 Graphic Cards. As shown, our method greatly decreases the training time to approximately 16% to 25% while achieving better performances, especially when there are mosaic samples with larger permutation counts. Please note that the time reduction is *ratio-based*, when larger datasets or models are trained, the absolute time reduction gap between baseline methods and our methods will be much more obvious.

4.2 Alleviating Memorizing

In the original instruction tuning process, each data sample will be trained several times for LLMs without changes to the instructions and responses. This training process poses risks to the potential memorizing effects on training samples, which can be partially indicated by the *"stair-like"* training loss curves as shown in Figure 3. In the figure, all the training settings are kept the same between the baseline models and Mosaic-IT models, including the Learning Rate, Warm-up Ratio, Learning Rate Schedule (Cosine), Batch Size, etc. For the baseline methods, the training loss hardly decreases within each epoch of training but drops dramatically when the LLMs meet the same training samples again, which indicates a potential memorizing effect of training samples and potential overfitting. However, when utilizing our method, the random mosaics of original instructions with diverse and complex meta-instructions largely diversify the overall training instructions. Although each original data sample will still be seen by LLMs several times during training, the overall context varies dramatically as each original sample is only an atomic element of the overall mosaic sample, indicating that there will be no identical overall instructions during the whole training process. Thus, this augmentation largely alleviates the potential memorizing and overfitting problems as shown in the figure, where the training loss decreases smoothly, representing the gradual learning process.

5 Why It Works

5.1 Preliminaries: Performance Degradation

The motivation of our Mosaic-IT is also rooted in the observation that when handling multiple instructions simultaneously, a performance degradation will incurred for even strong LLMs like GPT-4-turbo. While LLMs generally perform well when responding to single instructions, their capability to follow multiple instructions at once tends to decline noticeably. BatchPrompt has shown the uncertainty when LLMs are requested to answer multiple formatted questions at one time. Moreover, in some cases, e.g for general open-domain instructions, LLMs might directly ignore some of the instructions, especially when the LLMs are required to respond to the instructions in a random pre-defined order, which is exactly simulating our ***Permute*** strategy.

To quantitatively analyze this phenomenon, experiments using GPT-3.5-turbo and GPT-4-turbo are conducted on the WizardLM test set. Specifically, we compare the models' performance when responding to multiple instructions concurrently versus responding to a single instruction at each time, by utilizing LLM-based Pair-Wise comparison, as shown in Table 4. All the win rates are lower than 1.0, demonstrating a clear and significant reduction in response quality when these models are required to respond to multiple instructions at one time. Moreover, the possibility of missing instructions (Miss Rate) increases further when they are required to respond to the instructions in a predefined random order rather than a sequential order. These results clearly demonstrate the difficulties of following several instructions at a time and why it can be regarded as a higher level of instruction-following capability.

5.2 Further Analysis

> Mosaic-IT trains LLMs to follow meta-instructions for compositional reasoning.

Previous methods train LLMs to produce a response for a single instruction or query. Instead, our method produces a compositional data synthesis method to train LLMs to generate multiple responses for multiple instructions in diverse forms (e.g., order, mask, format) specified by different meta-instructions. It also enforces LLMs to partition the input context correctly and manage the interference and dependencies among multiple instructions. These are critical to developing and improving the compositional reasoning capabilities of LLMs, which have not been covered by mainstream instruction-tuning frameworks.

> Mosaic-IT creates more challenging and complex instructions to further improve LLMs' instruction-following capabilities.

Mosaic-IT's composition of multiple instructions and the diverse meta-instructions create more challenging and complex instruction-tuning data for LLMs. Moreover, since we do not rely on data synthesis using LLMs but solely apply some rules to existing data, the correctness and quality of the augmented data are guaranteed. As shown in Table 4, even powerful LLMs like GPT4 can not follow concatenated instructions. Besides, it has been widely accepted that such challenging and complex instructions improve LLMs' instruction-following capability (Zhao et al., 2024; Wu et al., 2024; Ding et al., 2023; Li et al., 2023a; Liu et al., 2023a; Li et al., 2024c,b; Guo et al., 2024). Mosaic-IT follows this intuition by making the instruction more challenging and complex in order to improve LLMs. Different from previous methods relying on humans or stronger teacher LLMs to create the challenging samples, Mosaic-IT does not require any humans/models to create the augmentations.

To quantitatively evaluate the difficulty and complexity of instruction-tuning data, InsTag (Lu et al., 2023) proposes a ChatGPT-based method (Number of InsTag), while Cherry (Li et al., 2024e) proposes a perplexity-based Instrutcion-Following Difficulty (IFD) score. We compute these two metrics on the Alpaca and WizardLM70k datasets to verify the effectiveness of our method in improving the difficulty/complexity:

Number of InsTag: The number of InsTag is used to measure the complexity of the instructions.

Pair-Wise (Multi vs. Single)	3 Instructions Win Rate ↑	3 Instructions Miss Rate ↓	5 Instructions Win Rate ↑	5 Instructions Miss Rate ↓	7 Instructions Win Rate ↑	7 Instructions Miss Rate ↓
GPT-3.5-turbo (Sequential)	0.357	0.014	0.336	0.055	0.303	0.064
GPT-3.5-turbo (Random)	0.315	0.124	0.330	0.156	0.198	0.312
GPT-4-turbo (Sequential)	0.176	0.000	0.137	0.000	0.140	0.000
GPT-4-turbo (Random)	0.139	0.000	0.153	0.014	0.101	0.005

Table 4: Pair-wise win rate of performances when responding to multiple instructions concurrently versus responding to a single instruction each time, and miss rate when responding to multiple instructions concurrently. "3 Instructions" represents the setting where 3 random instructions are concatenated together for inference. "Sequential" and "Random" represents the setting where the models are asked to respond to each instruction sequentially, or in a random pre-defined order.

Method	Ins Tag		IFD	
	Alpaca	Wizard-70k	Alpaca	Wizard-70k
Original	2.62	4.20	0.60	0.67
Mosaic-IT	**9.75**	**10.93**	**0.76**	**0.79**

Table 5: The comparison between the original dataset and Mosaic-IT enhanced dataset with respect to the **Number of InsTag** and the **IFD score**.

A larger value of the Number of InsTag indicates the intentions of the instruction are complex and benefit the LLM instruction tuning process. For the experiments below, we prompt GPT4o with the exact prompt provided in the paper to generate the Instags. As shown in Table 5, Mosaic-IT largely increases the average number of InsTag, indicating a large increase in instruction intention complexity, further leading to better performance.

IFD score: IFD score is a perplexity-based metric used to evaluate the instruction-following difficulty of a given instruction-response pair. A higher IFD score indicates that it is hard for the current model to build a connection between the instruction and the corresponding response, so it can be used to select training data beneficial for LLM instruction tuning. For the experiments below, we utilized the IFD score computed on GPT2. As shown in Table 5, Mosaic-IT increases IFD scores, indicating an increase in the instruction-following difficulty, which leads to an improvement in performance.

5.3 Qualitative Example

To better understand the differences between the responses generated by the baseline model and the Mosaic-IT model, a pair of qualitative examples are presented in Figure 4 (baseline model) and Figure 5 (Mosaic-IT model). The example instruction is selected from the WizardLM test sets and the models are Mistral-7B trained on Alpaca-GPT4 with or without our method.

Some of the real-world instructions can be complex and hard to answer all at once but require LLMs to "decompose" the original overall instruction into pieces to respond. Due to the lack of this kind of difficult data in the training set, the capability of most LLMs is largely restricted as shown in Figure 4. The baseline LLM directly ignores the "sub-query" of explaining "*protocols and standards such as TCP/IP, HTTP, FTP, DNS, DHCP, and ARP*". On the contrary, as a cost-free data synthesis method, Mosaic-IT simulates this kind of instruction by using original easy instructions, thus equipping LLMs with the capability to respond to these difficult instructions without ignoring some parts of it, as shown in Figure 5.

The better performances on Alpaca Eval 2 (LC) show that the performance gains of Mosaic-IT do not purely originate from longer responses. On the contrary, Mosaic-IT actually makes the lengths of training data more diverse compared with the original data, e.g., some of the samples are the concatenation of only 1 original sample, while some of them are the concatenation of 10 samples. This ensures diverse generation lengths according to the given instructions rather than being dominated by the original length distribution.

6 Conclusion

We introduce Mosaic Instruction Tuning (Mosaic-IT), a novel, human/model-free method to enhance instruction tuning for LLMs. By concatenating multiple instruction-response samples and using higher-level meta-instructions, Mosaic-IT improves multi-step and format-following capabilities. Our evaluations show superior performance and an 80% reduction in training costs compared to the original methods. Mosaic-IT's simplicity and efficiency make it a scalable solution for improving LLMs without extensive human intervention or resource-intensive teacher models. Our results highlight the potential of innovative data synthesis techniques in advancing LLM capabilities.

Limitations

The potential limitations of our work: (1) Currently, three Advanced Mosaic Strategies with corresponding high-level rules are proposed and utilized in our method, however, we believe more strategies and predefined rules can be further introduced. (2) The optimal distribution of the number of instructions for the mosaic process still needs further justification in future studies. (3) It is unknown whether the inclusion of extra models or careful curation/selection of instructions for concatenation will further improve the performance of Mosaic-IT largely.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. Yolov4: Optimal speed and accuracy of object detection. *Preprint*, arXiv:2004.10934.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Alexander Bukharin and Tuo Zhao. 2023. Data diversity matters for robust instruction tuning. *Preprint*, arXiv:2311.14736.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2023. Alpagasus: Training a better alpaca with fewer data. *Preprint*, arXiv:2307.08701.

Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. Batch prompting: Efficient inference with large language model APIs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 792–810, Singapore. Association for Computational Linguistics.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, Toronto, Canada. Association for Computational Linguistics.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Preprint*, arXiv:2205.14135.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Sivan Doveh, Assaf Arbelle, Sivan Harary, Roei Herzig, Donghyun Kim, Paola Cascante-Bonilla, Amit Alfassy, Rameswar Panda, Raja Giryes, Rogerio Feris, Shimon Ullman, and Leonid Karlinsky. 2023. Dense and aligned captions (DAC) promote compositional reasoning in VL models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Qianlong Du, Chengqing Zong, and Jiajun Zhang. 2023. Mods: Model-oriented data selection for instruction tuning. *Preprint*, arXiv:2311.15653.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335, Dublin, Ireland. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, and Ishan Misra. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *Preprint*, arXiv:2305.14387.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021. A framework for few-shot language model evaluation.

Weidong Guo, Jiuding Yang, Kaitong Yang, Xiangyang Li, Zhuwei Rao, Yu Xu, and Di Niu. 2024. Instruction fusion: Advancing prompt evolution through hybridization. *Preprint*, arXiv:2312.15692.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Miyoung Ko, Jinhyuk Lee, Hyunjae Kim, Gangwoo Kim, and Jaewoo Kang. 2020. Look at the first sentence: Position bias in question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1109–1121, Online. Association for Computational Linguistics.

Ming Li, Han Chen, Chenguang Wang, Dang Nguyen, Dianqi Li, and Tianyi Zhou. 2024a. Ruler: Improving llm controllability by rule-based data recycling. *arXiv preprint arXiv:2406.15938*.

Ming Li, Jiuhai Chen, Lichang Chen, and Tianyi Zhou. 2024b. Can LLMs speak for diverse people? tuning LLMs via debate to generate controllable controversial statements. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 16160–16176, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Jiuxiang Gu, and Tianyi Zhou. 2024c. Selective reflection-tuning: Student-selected data recycling for LLM instruction-tuning. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 16189–16211, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, and Tianyi Zhou. 2023a. Reflection-tuning: Recycling data for better instruction-tuning. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Ming Li, Yanhong Li, Ziyue Li, and Tianyi Zhou. 2025. How instruction and reasoning data shape post-training: Data quality through the lens of layer-wise gradients. *arXiv preprint arXiv:2504.10766*.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024d. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14255–14273, Bangkok, Thailand. Association for Computational Linguistics.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024e. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7595–7628, Mexico City, Mexico. Association for Computational Linguistics.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. 2023b. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023c. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Jianzhe Lin, Maurice Diesendruck, Liang Du, and Robin Abraham. 2024. Batchprompt: Accomplish more with less. In *The Twelfth International Conference on Learning Representations*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2023a. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023b. G-eval: Nlg evaluation using gpt-4 with better human alignment. *Preprint*, arXiv:2303.16634.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models. *Preprint*, arXiv:2308.07074.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *Preprint*, arXiv:2304.03277.

Teven Le Scao, Angela Fan, Christopher Akiki, Elizabeth-Jane Pavlick, Suzana Ili'c, Daniel Hesslow, Roman Castagn'e, Alexandra Sasha Luccioni, Franccois Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Rose Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurenccon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa Etxabe, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris C. Emezue, Christopher Klamm, Colin Leong, Daniel Alexander van Strien, David Ifeoluwa Adelani, Dragomir R. Radev, Eduardo Gonz'alez Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady ElSahar, Hamza Benyamina, Hieu Trung Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jorg Frohberg, Josephine L. Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro von Werra, Leon Weber, Long Phan, Loubna Ben Allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, Mar'ia Grandury, Mario vSavsko, Max Huang, Maximin Coavoux, and Mayank Singh. 2022. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv*, abs/2211.05100.

Andrea Sottana, Bin Liang, Kai Zou, and Zheng Yuan. 2023. Evaluation metrics in the era of GPT-4: Reliably evaluating large language models on sequence to sequence tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8776–8788, Singapore. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023a. Large language models are not fair evaluators. *Preprint*, arXiv:2305.17926.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. 2024. Lamini-lm: A diverse herd of distilled models from large-scale instructions. *Preprint*, arXiv:2304.14402.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *Preprint*, arXiv:2304.12244.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng

Tao, and Tianyi Zhou. 2024a. A survey on knowledge distillation of large language models. *ArXiv*, abs/2402.13116.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2024b. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *Preprint*, arXiv:2406.08464.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. CrossFit: A few-shot learning challenge for cross-task generalization in NLP. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7163–7189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2024. Evaluating large language models at evaluating instruction following. In *The Twelfth International Conference on Learning Representations*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *Preprint*, arXiv:2303.18223.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Fei Huang, Yongbin Li, and Nevin L. Zhang. 2024. A preliminary study of the intrinsic relationship between complexity and alignment. *Preprint*, arXiv:2308.05696.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. 2024a. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *Preprint*, arXiv:2309.11998.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024b. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. *Preprint*, arXiv:2305.11206.

A Qualitative Examples

To better understand the differences between the responses generated by the baseline model and the Mosaic-IT model, a pair of qualitative examples are presented in Figure 4 (baseline model) and Figure 5 (Mosaic-IT model).

Identify some of the main components of a computer network such as hosts, routers, switches, hubs, links, and interfaces. Explain how they communicate and coordinate with each other using protocols and standards such as TCP/IP, HTTP, FTP, DNS, DHCP, and ARP. Describe how data is transmitted and received over a network using concepts such as packets, frames, headers, addresses, ports, sockets, and checksums.

A computer network is composed of several components that work together to facilitate communication and data transfer between devices. These components include:
1. Hosts: A host is a device that is connected to the network, such as a computer, smartphone, or printer. Each host has a unique IP address, which is used to identify it on the network.
2. Routers: A router is a device that connects multiple networks together and directs traffic between them. It uses protocols such as IP to route data packets to their destination.
3. Switches: A switch is a device that connects multiple devices together on a single network. It uses MAC addresses to direct data packets to their destination.
4. Hubs: A hub is a device that connects multiple devices together on a single network. It broadcasts data packets to all connected devices, which allows them to communicate with each other.
5. Links: A link is the physical connection between two devices on a network. It can be a cable, a wireless connection, or a combination of both.
6. Interfaces: An interface is the part of a device that connects it to the network. It can be a network card, a USB port, or a wireless card.

All of these components communicate and coordinate with each other using protocols and standards such as TCP/IP, HTTP, FTP, DNS, DHCP, and ARP. These protocols define how data is transmitted and received over a network, how devices are addressed, and how traffic is routed.
Data is transmitted over a network in packets or frames. Each packet or frame contains a header, which includes information such as the destination IP address, the source IP address, and the type of data being transmitted. The data is then transmitted over the network, and the receiving device uses the information in the header to identify the data and direct it to the appropriate application or program.

In summary, a computer network is composed of several components that work together to facilitate communication and data transfer. These components communicate and coordinate with each other using protocols and standards, and data is transmitted over the network in packets or frames, using headers to identify the destination, source, and type of data.

Figure 4: An example from the WizardLM test set with the response generated by the Baseline model.

A computer network is made up of several components that work together to facilitate communication and data transfer between devices. Some of the main components include:
1. Hosts: These are the devices that are connected to the network and use its resources, such as computers, printers, and servers.
2. Routers: Routers are devices that direct traffic between networks, and are used to connect multiple networks together.
3. Switches: Switches are used to direct traffic within a network, and are used to connect multiple devices together on a single network.
4. Hubs: Hubs are devices that connect multiple devices together on a single network, and are used to distribute data to multiple devices at once.
5. Links: Links are the physical connections between devices on a network, such as Ethernet cables or wireless connections.
6. Interfaces: Interfaces are the points of connection between the network and the devices on it, such as network cards or wireless adapters.

For these components to communicate and coordinate with each other, they use a variety of protocols and standards. Protocols are sets of rules that dictate how data is transmitted and received, while standards are agreed-upon methods or formats that allow for interoperability between different systems. Some common protocols and standards used in computer networking include:
- TCP/IP: This is a suite of protocols that is used to communicate between devices on a network. It includes protocols for addressing, routing, and transporting data.
- HTTP: This is a protocol used for accessing and transferring data over the web. It is used to communicate between web servers and clients.
- FTP: This is a protocol used for transferring files over a network. It is used to communicate between FTP servers and clients.
- DNS: This is a protocol used for translating domain names to IP addresses. It is used to coordinate communication between devices on a network.
- DHCP: This is a protocol used for assigning IP addresses to devices on a network. It is used to coordinate communication between devices on a network.
- ARP: This is a protocol used for finding the MAC address of a device on a network. It is used to coordinate communication between devices on a network.

When data is transmitted over a network, it is broken down into small packets, each of which contains information about where it needs to go and what it contains. These packets are sent over the network, hopping from one device to the next until they reach their destination. At each hop, the device examines the packet's header to see where it needs to go next, and then forwards it along the appropriate link.
Once the packet reaches its destination, the receiving device examines the header to see if it is the intended recipient. If it is, the device will extract the data from the packet and use it, otherwise, the packet will be discarded.
To ensure that the data in the packets is not corrupted during transmission, checksums are used. A checksum is a mathematical calculation performed on the data in the packet, which produces a value that can be used to verify that the data has not been altered during transmission. If the checksum of the data in a packet is incorrect when it arrives at its destination, it is discarded, and the sender is notified of the error.

In summary, data is transmitted over a network by being broken down into small packets, each of which contains information about where it needs to go and what it contains. These packets are sent hopping from one device to the next, using a variety of protocols and standards to coordinate their transmission and reception. Checksums are used to ensure that the data in the packets has not been corrupted during transmission, and any errors are detected and corrected.

Figure 5: An example from the WizardLM test set with the response generated by the Mosaic-IT model.

B Experimental Setup

B.1 Implementation Details

For the three base pre-trained models, Llama2-7B, Llama2-13B (Touvron et al., 2023b), and Mistral-7B (Jiang et al., 2023), we utilize the prompt and code base from Vicuna (Chiang et al., 2023) and flash attention (Dao et al., 2022). The overall training arguments are aligned with the common training configuration. The Adam optimizer (Kingma and Ba, 2017) is utilized with the batch size of 128 and with the max token length of 2048. When training the baseline models Llama2-7B and Llama2-13B, the maximum learning rate is set to 2×10^{-5} with the warmup rate as 0.03 for 3 epochs. When training the baseline models on Mistral-7B, the maximum learning rate is set to 1×10^{-5} with the warmup rate as 0.1 for 3 epochs. For the three models, Llama-3-8B (Dubey et al., 2024), Phi-3 (Abdin et al., 2024), and Gemma2-2B (Team et al., 2024), we utilize the code base from LLaMA-Factory (Zheng et al., 2024b). The max token length is set with 4096 following the modern settings and we train the model for 2 epochs. Other parameters are kept the same as the above.

When training with Mosaic-IT, we run the mosaic process n times for each experiment to simulate n epochs of training, n represents the number of epochs trained on baseline models, to ensure the alignment of overall data sample counts. Then these augmented data are mixed together and used for training 1 epoch while all other configurations are kept the same as baselines.

B.2 Training Dataset

The Alpaca dataset (Taori et al., 2023) comprises $52,000$ instruction-following samples and is constructed utilizing the self-instruct paradigm (Wang et al., 2023b). This dataset was produced by employing OpenAI's text-davinci-003 model. Characterized as a classical dataset with moderate quality attributes, the Alpaca dataset serves as an initial platform to validate our methodology. To further substantiate our approach using a dataset of inherently high quality, we also applied our method to the Alpaca-GPT4 dataset (Peng et al., 2023), which features responses generated by GPT4. The WizardLM dataset (Xu et al., 2023) is also utilized in our method, which contains $70,000$ samples created by the evolution algorithm proposed by them. With ChatGPT-3.5 utilized, the data quality on WizardLM is largely guaranteed. The Vicuna 1M dataset (Zheng et al., 2024a) is a large-scale dataset containing one million real-world conversations with 25 state-of-the-art LLMs. Due to the computation budget, 300k instances are randomly sampled for our experiments. Magpie dataset (Xu et al., 2024b) is the most recent SOTA synthetic dataset with 300k samples.

B.3 Evaluation Metrics

Pair-wise Comparison by using powerful LLMs like GPT-4 is recently widely accepted and becoming a common practice (Touvron et al., 2023b; Chiang et al., 2023; Dettmers et al., 2023; Liu et al., 2023b; Chiang and Lee, 2023). The evaluation of responses from LLMs, especially in open-domain contexts where definitive ground truth is hard to establish, continues to be an intricate and evolving research domain. Recent studies, however, have indicated a notable alignment between GPT-4's performance evaluations and human assessments (Zheng et al., 2023; Li et al., 2023c; Sottana et al., 2023), thereby establishing a credible foundation for this evaluative methodology. We adopted test instruction sets from WizardLM (Xu et al., 2023), comprising 218 diverse, human-curated instructions for pair-wise comparison. We directly follow the evaluation framework proposed by (Chen et al., 2023; Li et al., 2024e), which evaluates responses on a scale spanning from 1 to 10 across multiple dimensions. To further address positional bias, as discussed by (Ko et al., 2020; Wang et al., 2023a), the comparison is conducted in two distinct sequences, LLM1's response first and then LLM2's response first, ensuring a fair assessment of model performance. Evaluation outcomes are categorized into 'win-tie-loss' for each instruction. The detailed evaluation prompt is provided in Appendix H.

Open LLM Leaderboard, employing the evaluation framework from Eval Harness (Gao et al., 2021), offers a detailed and systematic approach to assessing the capabilities of generative language models through a set of diverse evaluation tasks. This methodology zeroes in on four pivotal benchmarks: ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), and TruthfulQA (Lin et al., 2022). These benchmarks collectively provide a comprehensive evaluation of the models' reasoning abilities, their grasp of common-sense knowledge, and their accuracy in presenting factual information. Consequently, the leaderboard presents valuable insights.

Alpaca-Eval Leaderboard, leveraging the AlpacaFarm evaluation dataset, presents a dependable and efficient automated evaluation tool for LLMs (Li et al., 2023c; Dubois et al., 2023). This tool benchmarks the responses generated by LLMs against those from Davinci003, focusing on the models' ability to adhere to generic user instructions.

MT-Bench (Multi-turn Benchmark) (Zheng et al., 2023) is a benchmark tool designed for automated evaluating LLMs in multi-turn dialogue settings. It focuses on analyzing conversation flow and the model's ability to follow instructions with 80 high-quality, multi-turn questions.

IFEval (Instruction-Following Eval) (Zeng et al., 2024) is a straightforward and easy-to-produce evaluation benchmark focusing on a set of "verifiable instructions". It contains 25 types of verifiable instructions and 541 prompts, with each prompt containing one or multiple verifiable instructions.

Human Evaluation is further implemented to substantiate the superiority of our approach based on the WizardLM test set. The test set contains 100 samples randomly sampled from the original WizardLM test set. Three human evaluators were tasked with comparing the outputs generated by the models under consideration, using the same criteria as in the previous pairwise evaluation. Each evaluator was presented with three response options: Win, Tie, and Loss. The final outcomes were determined by a majority vote.

C Detailed Ablation Studies

In this section, extensive ablation experiments are conducted on Mistral-7B using the Alpaca-GPT4 dataset to verify our method. We utilize Pair-wise comparison for evaluation.

C.1 Ablation on Mosaic Strategies

Ablation on Mosaic Strategies is presented in Table 6a. *"**Primary**"* represents the Primary Mosaic Strategy. The winning score of this setting is greater than 1.0, indicating a better performance compared with the baseline method. This comparison directly verifies the effectiveness of the idea of introducing multiple instructions during training, which complicates the instructions at no cost and improves the instruction-following ability of LLMs. *"**Format**"* represents the Format Strategy. Although the winning score is only slightly greater than the naive version, this version makes it possible for LLMs to follow the customized user-defined formats, indicating great potential for the controllability of LLMs. Moreover, the format version can be easily used with other types of meta instructions, showing great extensibility. *"**Permute**"* represents the Permute Strategy that builds on the Format Strategy with a probability of 1/2, similar to *"**Maskout**"*. *"**Permute/Maskout**"* represents our default setting, where the Permute or Maskout Strategies are utilized together with the Format Strategy with a probability of 1/3. All these 3 settings show higher performance than the format version, indicating the effectiveness of Advanced Mosaic Strategies, which define more complicated meta instructions.

C.2 Ablation on the Max Number of Instructions

Ablation on the Max Number of Instructions is presented in Table 6b, including the pair-wise comparison values. As shown in the table, when the max number is set as 2, i.e., at most 2 instructions/responses are concatenated together, the performance is almost the same as the baseline, indicating the ineffectiveness. However, when the max number grows, the corresponding winning scores also grow consistently. This trend shows that the more instructions concatenated together, the better the instruction-following ability. We hypothesize that, with the growth of the number of instructions, the overall instruction becomes much harder to follow, especially for the permute and maskout strategies, which benefit LLMs' instruction-following capability.

	Winning Score	Win	Tie	Lose
Primary	1.261	110	55	53
Format	1.284	109	62	47
Permute	1.334	118	55	45
Maskout	1.376	121	58	39
Permute/Maskout	1.349	123	48	47

(a) Ablation on Mosaic-IT strategies.

	Winning Score	Win	Tie	Lose
Max Count = 2	0.989	70	75	73
Max Count = 4	1.142	92	65	61
Max Count = 6	1.303	111	62	45
Max Count = 8	1.294	112	58	48
Max Count = 10	1.349	123	48	47
Max Count = 12	1.376	124	52	42

(b) Ablation on the Max Number of Instructions.

Table 6: Ablation on (a) Mosaic-IT strategies and (b) Max Number of Instructions.

	Winning Score	Win	Tie	Lose	Mix ≤ 5
Fix	0.982	90	34	94	2.39%
Exponential	0.995	94	29	95	2.58%
Pareto	1.417	129	51	38	8.94%
Log-normal	1.431	136	40	42	6.83%
Logistic	1.417	123	49	46	15.84%
Uniform	1.349	123	48	47	51.45%

Table 7: Ablation on the **Distribution of Number of Instructions**. The distribution formula and data counts for different settings are shown in Appendix E. "Mix $\leq$ 5" represents the percentage of samples with the number of instructions less or equal to 5.

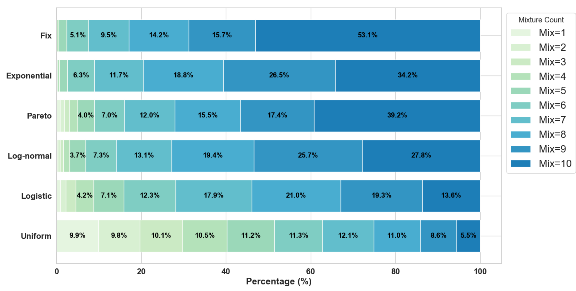


Figure 6: Ablation on the **Distribution of Number of Instructions**, the visualization of distributions.

C.3 Ablation on the Distribution of Number of Instructions

Ablation on the Distribution of Number of Instructions is presented, including the pair-wise comparison values in Table 7 and detailed number distribution comparisons in Figure 6, which aims at identifying how this count distribution affects the performance of our method. The detailed distribution formula and data counts are provided in the Appendix E. *"**Fix**"* represents the setting where all the overall instructions are concatenated with a fixed number of instructions, which we set as 10 unless the overall instructions exceed the max length limit. *"**Exponential**"* represents the setting where the number of instructions is sampled following the exponential distribution. Under these two settings, less than 3% of the overall instructions are concatenated by less than or equal to 5 original instructions. The lack of few-instruction concatenated samples negatively affects the LLMs' ability to follow the single instruction, which is employed by most of the existing evaluation methods, leading to worse performance. *"**Pareto**"*, *"**Log-normal**"*, and *"**Logistic**"* represents the corresponding distribution that are utilized for sampling. Different from the above two settings, approximately 10% of the overall instructions are composed of fewer original instructions, thus ensuring the LLMs are trained with samples with sufficiently diverse lengths, resulting in optimal performances. *"**Uniform**"* is our default setting, representing using the uniform distribution where different numbers are sampled evenly. In this situation, the LLMs are trained with samples with the most diverse lengths, thus avoiding the LLMs overfitting to simple lengthy responses.

C.4 Ablation on Semantic Grouping

To demonstrate that including other concatenation methods could further strengthen our argument, we further conduct experiments using a semantic grouping approach based on the Alpaca-GPT4 dataset to fine-tune Mistral.

Specifically, we utilize a sentence transformer model *all-mpnet-base-v2*[1] to obtain the semantic embedding for each sample in the dataset, and then we apply the K-means algorithm to group these data samples into multiple clusters. To ensure enough samples per cluster, we set K=52 as the dataset contains 52k samples in total. Given the clusters, each concatenated sample is composed of multiple samples randomly drawn from the same cluster. We keep using the same training hyperparameters as in our main experiments. As shown in Table 8, we report the performance on two evaluation metrics: pair-wise comparison and Alpaca Eval.

The semantic concatenation can still outperform the non-mosaic baseline by a large margin, indicating the effectiveness and potential of our Mosaic-IT augmentations and tasks. The semantic concatenation method has a slightly lower performance than the pure-random concatenation method, on pair-wise comparison and Alpaca Eval 2 scores. However, it achieves a much higher Alpaca Eval 2 (LC) score. This result suggests that the response quality of the model trained with semantic concatenation is on par with pure-random, but the response length is shorter and more condensed. We found that semantic grouping leads to clusters with highly different average lengths of samples: The largest average length is 316.7 tokens, while the smallest is 31.4 tokens. This discrepancy makes the lengths of Mosaic-IT concatenated samples more diverse, resulting in a better trade-off between the quality and length of the responses.

[1] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

Method	Alpaca Eval 2 (LC)	Alpaca Eval 2	Pair-wise Compare (with non-mosaic)	Pair-wise Compare (with pure-random)
Pure-random Concatenation	5.00	**7.81**	**1.349**	**1.000**
Concatenation with Semantic Groups	**7.80**	6.51	1.275	0.936

Table 8: Comparison with Semantic grouping

D Multi-Instruction Evaluation

To verify our trained models' capability to follow multiple instructions and meta-instructions in one inference, we create a test set of compositional instructions from WizardLM test sets using Mosaic-IT. For simplicity, we name this new test setting as *Mosaic Task*, which evaluates LLMs' capability to follow multiple instructions with additional diverse constraints (meta-instructions). One example of *Mosaic Task* is shown as follows.

Example of Mosaic Task

System Prompt
You are a helpful and precise assistant for providing the answer.

User Prompt
Respond to each of the following instructions in reverse of the original order.
[Ins1]
[Ins2]
[Ins3]

Figure 7: The prompt we used to request GPT4-Turbo to evaluate the responses.

We use the success rate(%) to evaluate the performance of models on the Mosaic task. A response is successful if it follows the meta-instruction and no instruction is ignored (unless the meta-instruction masks it). In the table below, we report the success rate (%) of LLMs following three meta-instruction strategies, i.e., Format, Permute, and Maskout, on compositional augmentations of different numbers of instructions (i.e., 3, 5, 7 instructions). We report the success rates of GPT4o, two base models, and their Mosaic-IT finetuned versions, as shown in Table 9.

The results expose the weaknesses of existing LLMs on Mosaic-IT tasks and show that training on Mosaic-IT augmentations can significantly improve performance. Specifically, existing LLMs, even GPT4o, can not perfectly follow multiple instructions with diverse constraints, not to mention other open-source models like Llama3 finetuned on datasets such as Magpie. These results further demonstrate the difficulty and complexity of Mosaic-IT tasks for existing LLMs, indicating the novelty of our method. The compositional reasoning capability required by Mosaic-IT tasks cannot be covered by the capabilities of base LLMs and existing instruction-tuning datasets. For example, the success rates of Mistral + Alpaca-GPT4 (baseline) and Llama3 + Magepie (baseline) are similar, although Llama3 + Magepie has relatively better general instruction-following capabilities among them.

Our method can bridge the significant gap and enhance LLMs' capability to follow multiple instructions with diverse constraints. Moreover, our data augmentation is cost-free and does not take any effort from humans or models.

Model	3 Instructions			5 Instructions			7 Instructions		
	Format	Permute	Maskout	Format	Permute	Maskout	Format	Permute	Maskout
GPT4o	59.17	55.05	41.46	56.88	51.38	26.13	29.82	37.16	24.27
Mistral + Alpaca-GPT4 (baseline)	20.18	3.67	3.25	10.09	2.75	5.41	7.34	0.92	0.97
Mistral + Alpaca-GPT4 (mosaic)	98.32	66.51	69.11	95.87	60.55	67.57	97.25	64.68	66.02
Llama3 + Magepie (baseline)	16.06	8.26	7.32	9.63	1.38	5.41	5.50	2.75	3.88
Llama3 + Magepie (mosaic)	97.71	79.82	84.55	94.95	72.94	77.48	76.61	61.01	85.44

Table 9: Performance comparison across multiple instruction settings.

E Detailed Distribution for Ablation on Mixture Distribution

E.1 Distribution description

The detailed distribution descriptions and formulas are provided below.

Exponential Distribution[2]: The exponential distribution is a continuous probability distribution used to model the time or space between events in a Poisson process. The probability density function (PDF) of the exponential distribution is:

$$f(x; \lambda) = \lambda e^{-\lambda x} \quad \text{for } x \geq 0,$$

where $\lambda = 1$ by default in our setting. We will resample with this distribution if the sampled value x_{sample} is greater than k_{max}.

Log-normal Distribution[3]: The log-normal distribution is a continuous probability distribution of a random variable whose logarithm is normally distributed. It is often used to model variables that are positively skewed, such as income, stock prices, and other financial data. The probability density function (PDF) for a log-normal distribution is given by:

$$f(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln x - \mu)^2}{2\sigma^2}\right) \quad \text{for} \quad x > 0$$

where $\mu = 0$ and $\sigma = 0$ by default in our setting. We will resample with this distribution if the sampled value x_{sample} is greater than k_{max}.

Logistic Distribution[4]: The logistic distribution is a continuous probability distribution used in various fields, including logistic regression, modeling growth, and in some cases as an alternative to the normal distribution due to its heavier tails. The probability density function (PDF) for the logistic distribution is given by:

$$f(x; \mu, s) = \frac{e^{-(x-\mu)/s}}{s\left(1 + e^{-(x-\mu)/s}\right)^2}$$

where $\mu = 0$ and $s = 2$ by default in our setting. We will resample with this distribution if the sampled value x_{sample} is greater than k_{max}.

Pareto Distribution[5]: The Pareto II or Lomax distribution is a shifted Pareto distribution. It can be considered a simplified version of the Generalized Pareto distribution, with the scale set to one and the location set to zero. The probability density function (PDF) for the Pareto distribution is:

$$f(x; \alpha) = \frac{\alpha m^\alpha}{x^{\alpha+1}} \quad \text{for} \quad x \geq m,$$

where $m = 1$ and $\alpha = 1$ by default in our setting. We will resample with this distribution if the sampled value $x_{sample} - 1$ is greater than k_{max}.

After getting x_{sample}, a floor function will be utilized to get the corresponding integer and the final concatenation count $k = k_{max} - floor(x_{sample})$.

E.2 Distribution visualization

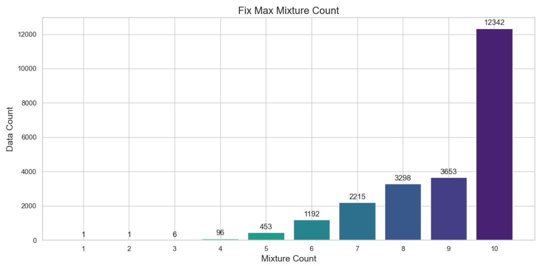
The detailed data counts for different distributions are provided in Figure 8.

[2] https://numpy.org/doc/stable/reference/random/generated/numpy.random.exponential.html

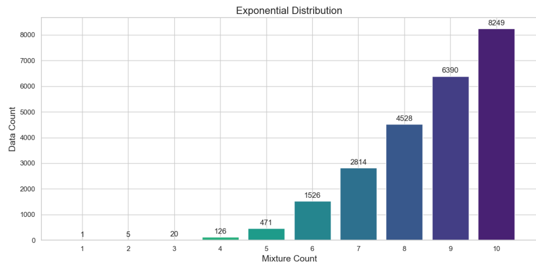
[3] https://numpy.org/doc/stable/reference/random/generated/numpy.random.lognormal.html

[4] https://numpy.org/doc/stable/reference/random/generated/numpy.random.logistic.html

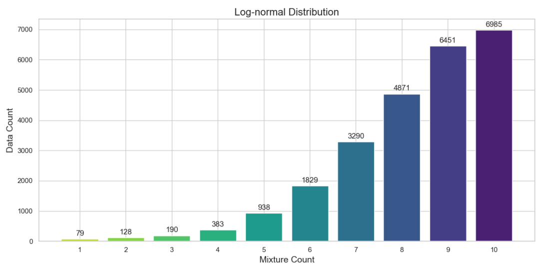
[5] https://numpy.org/doc/stable/reference/random/generated/numpy.random.pareto.html



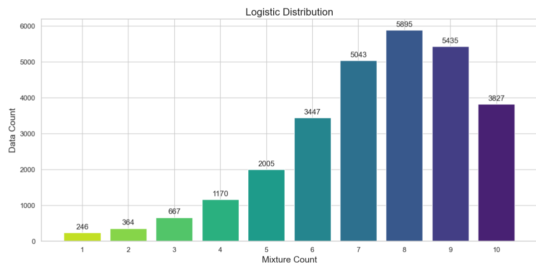
(a) Fix Max Number



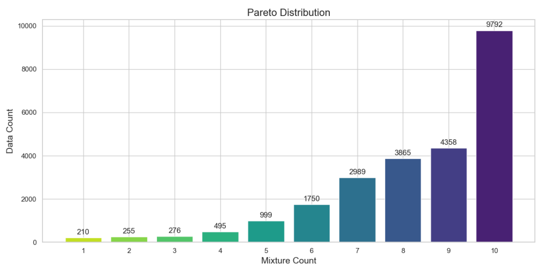
(b) Exponential Distribution



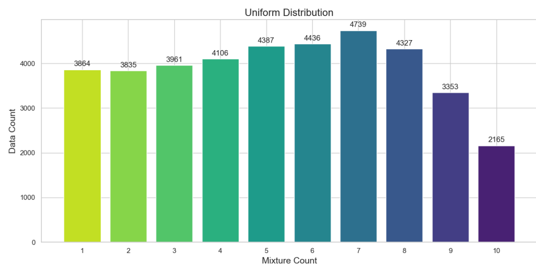
(c) Log-normal Distribution



(d) Logistic Distribution



(e) Pareto Distribution



(f) Uniform Distribution

Figure 8: Bar plots of detailed data counts for different distributions in the Ablation on the Numbers of Instructions: (a) Fix Max Number, (b) Exponential Distribution, (c) Log-normal Distribution, (d) Logistic Distribution, (e) Pareto Distribution, (f) Uniform Distribution.

F Related Work

Earlier research in instruction tuning primarily centered on constructing expansive, high-quality datasets through intensive curation by human experts, a process both time-consuming and labor-intensive (Khashabi et al., 2020; Ye et al., 2021; Wei et al., 2022; Wang et al., 2022; Du et al., 2022). Motivated by the success of Alpaca (Taori et al., 2023), recent studies have explored automated approaches for developing instruction-tuning datasets.

Instruction Data Improvement: WizardLM (Xu et al., 2023) first proposes an Evol Algorithm to complicate the existing data and reach supreme performance. LaMini-LM (Wu et al., 2024) innovatively generates "Topic-Guided" instructions utilizing Wiki data. Tree-Instruct (Zhao et al., 2024) preliminarily explores the relationship between instruction complexity and Alignment and proposes adding nodes to complicate the instruction. UltraChat (Ding et al., 2023) establishes broad thematic scopes, systematically generating numerous instructions within each. Reflection-Tuning (Li et al., 2023a) sequentially refines both instructions and responses by focusing on specific evaluative criteria. DEITA (Liu et al., 2023a) utilizes ChatGPT to diversify and then select the data. Selective Reflection-Tuning (Li et al., 2024c) proposes a teacher-student collaborative pipeline to improve and select the data. Instruction Fusion (Guo et al., 2024) proposes to utilize ChatGPT4 to merge two distinct instructions for further complexity enhancement. These advancements showcase a shift towards automating the generation and refinement of datasets, reducing reliance on human labor.

Instruction Data Selection: It is widely accepted that "quality is all you need" (Touvron et al., 2023b; Zhou et al., 2023) for instruction tuning. LIMA (Zhou et al., 2023) demonstrates that merely 1,000 human-carefully-curated, high-quality training instances can substantially enhance the instruction-following performance. InsTag (Lu et al., 2023) employs the proprietary model, ChatGPT, to tag instruction data and select data with complex tags. Alpagasus (Chen et al., 2023) utilizes proprietary LLMs chatGPT and Claude2 to directly assess the quality of instruction tuning data. Cherry LLM (Li et al., 2024e) proposes the Instruction-Following Difficulty (IFD) scores to assess the difficulty of the instructions, which is a self-guided method in which no extra LLMs are utilized. Motivated by Humpback (Li et al., 2023b), Selective Reflection-Tuning (Li et al., 2024c) extends the IFD score to a reverse version, focusing on the feasibility of responses. (Du et al., 2023) and (Bukharin and Zhao, 2023) utilize reward models as the base scores for measuring data quality. DEITA (Liu et al., 2023a) experiments on several different data selection metrics and builds a dataset with high quality. Superfiltering (Li et al., 2024d) reveals the consistency between weak and strong language models in perceiving instruction difficulty, making the filtering process much more efficient. A recent work (Li et al., 2025) tries to unify the different data evaluation metrics through the lens of gradients. All these works are devoted to distinguishing and selecting good data samples from bad ones for instruction tuning.

G Predefined Rules

Examples of predefined formats can be found in Table 10 and detailed predefined rule descriptions can be found in Table 11.

<table>
<tr><th>Serial Digit</th><th>Parsing Bracket</th><th>Parsing Text</th><th>Assembled Examples</th></tr>
<tr><td>i</td><td>(text)</td><td>BEGIN, END</td><td>1. (BEGIN)response(END)</td></tr>
<tr><td>(i)</td><td>[text]</td><td>START, END</td><td>(1). [START]response[END]</td></tr>
<tr><td>[i]</td><td>⟨text⟩</td><td>RESPONSE, END</td><td>[1]. ⟨RESPONSE⟩response⟨END⟩</td></tr>
<tr><td>⟨i⟩</td><td>≪text≫</td><td>RESPONSE, END OF RESPONSE</td><td>⟨1⟩. ≪RESPONSE≫response≪END OF RESPONSE≫</td></tr>
<tr><td>≪i≫</td><td>|text|</td><td>OPEN, CLOSE</td><td>≪1≫. |OPEN|response|CLOSE|</td></tr>
<tr><td>###i</td><td>[|text|]</td><td>OPEN RESPONSE, CLOSE</td><td>###1. [|OPEN RESPONSE|]response[|CLOSE|]</td></tr>
<tr><td>##i</td><td>⟨|text|⟩</td><td>INITIATE, TERMINATE</td><td>##1. ⟨|INITIATE|⟩response⟨|TERMINATE|⟩</td></tr>
<tr><td>##i##</td><td>#text#</td><td>START POINT, END POINT</td><td>##1##. #START POINT#response#END POINT#</td></tr>
<tr><td>|i|</td><td>*text*</td><td>RES_START, RES_END</td><td>|1|. *RES_START*response*RES_END*</td></tr>
<tr><td>||i||</td><td>@text@</td><td>RES, /RES</td><td>||1||. @RES@response@/RES@</td></tr>
</table>

Table 10: Examples of predefined formats, including the Serial Digit formats and Response Parsing formats. "*i*" represents the real number serial number, "*text*" represents the replaceable parsing text, and "*response*" represents the real response of the concatenated overall instructions/responses. The response parsing formats are composed of the parsing bracket and text. In each mosaic process, random formats will be sampled simulating the real-world user-defined formats. The last column represents the assembled examples using the formats in the same row.

Strategy	Rule Name	Rule Description
Permute	FIX	Respond in the order of a provided list.
Permute	REVERSE	Respond in reverse of the original order.
Permute	ALPHA	Respond in the alphabetical order of the first letter of tasks.
Permute	REVERSE_ALPHA	Respond in the reverse alphabetical order of the first letter of tasks.
Permute	LENGTH_WORD	Respond according to the length (words) of tasks, respond to short ones first.
Permute	REVERSE_LENGTH_WORD	Respond according to the length (words) of tasks, respond to long ones first.
Permute	LENGTH_CHAR	Respond according to the length (characters) of tasks, respond to short ones first.
Permute	REVERSE_CHAR_WORD	Respond according to the length (characters) of tasks, respond to long ones first.
Permute	ODD_EVEN	First respond to the odd-numbered tasks, then the even-numbered ones.
Permute	EVEN_ODD	First respond to the even-numbered tasks, then the odd-numbered ones.
Maskout	FIX	Ignore the tasks provided in the list.
Maskout	WORD_LONG	Ignore the longest one/several task(s) according to the word count.
Maskout	WORD_SHORT	Ignore the shortest one/several task(s) according to the word count.
Maskout	ODD	Ignore the odd-numbered tasks.
Maskout	EVEN	Ignore the even-numbered tasks.

Table 11: Predefined rules for the Permute and Maskout strategy. A random rule will be sampled for each mosaic process, which largely complicates and diversifies the mosaicked instructions.

H Prompt for Evaluation

The detailed pair-wise comparison prompt for the pair-wise comparison is in Figure 9.

Prompt for Performance Evaluation

System Prompt
You are a helpful and precise assistant for checking the quality of the answer.

User Prompt
[Question]
Question
[The Start of Assistant 2's Answer]
Answer 2
[The End of Assistant 2's Answer]
[The Start of Assistant 2's Answer]
Answer 2
[The End of Assistant 2's Answer]

We would like to request your feedback on the performance of two AI assistants in response to the user question displayed above.
Please rate the helpfulness, relevance, accuracy, level of details of their responses. Each assistant receives an overall score on a scale of 1 to 10, where a higher score indicates better overall performance.
Please first output a single line containing only two values indicating the scores for Assistant 1 and 2, respectively. The two scores are separated by a space. In the subsequent line, please provide a comprehensive explanation of your evaluation, avoiding any potential bias and ensuring that the order in which the responses were presented does not affect your judgment.

Figure 9: The prompt we used to request GPT4-Turbo to evaluate the responses.

I Detailed Performance Scores on Llama3, Phi3 and Gemma2

The detailed performance scores on the Open LLM Leaderboard and IFEval, for Llama-3-8B, Phi-3, and Gemma2-2B.

Model	Dataset	Method	Open LLM Leaderboard ↑					IF Eval ↑			
			Average	ARC	HellaSwag	MMLU	TruthfulQA	Prompt (S)	Inst (S)	Prompt (L)	Inst (L)
Llama-3-8B	**Vicuna**	Baseline	52.51	44.54	70.66	49.68	45.18	19.04	30.70	21.26	33.45
		Mosaic-IT	**55.62**	47.78	73.77	56.11	44.83	**29.76**	**43.17**	**31.42**	**45.56**
	Magpie	Baseline	56.15	50.09	71.29	54.40	48.84	29.39	40.76	35.67	47.72
		Mosaic-IT	**60.13**	53.58	76.62	60.82	49.52	**38.08**	**49.64**	**40.67**	**52.76**
Phi-3	**Vicuna**	Baseline	62.06	58.96	76.48	64.89	47.89	28.47	**40.29**	30.50	**43.17**
		Mosaic-IT	**62.30**	58.45	77.66	65.24	47.87	**30.13**	39.57	**32.35**	41.85
	Magpie	Baseline	62.90	59.30	75.07	65.89	51.35	39.56	50.84	44.36	55.25
		Mosaic-IT	**63.54**	60.23	76.30	66.14	51.50	**42.33**	**53.60**	**50.83**	**62.35**
Gemma2-2B	**Vicuna**	Baseline	48.90	43.43	64.20	41.50	46.46	20.51	32.61	23.66	35.61
		Mosaic-IT	**51.31**	46.33	69.32	44.29	45.31	**21.44**	**33.57**	**24.03**	**36.93**
	Magpie	Baseline	46.37	39.59	60.71	35.46	49.75	19.78	29.74	21.81	32.49
		Mosaic-IT	**48.36**	39.33	64.10	39.87	50.16	**19.78**	**31.65**	**22.18**	**34.77**

Table 12: The performance comparison on more model families and datasets on all five automatic evaluation metrics. In IF Eval, P and I represent Prompt-level and Instruction-level accuracy.